DTIC FILE COPY

4

ISI Research Report
ISI/RR-89-199
July 1989

AD-A211 288

University of Southern California

Robert MacGregor
John Yen

The Knowledge Representation Project

89 8 [illegible]

INFORMATION SCIENCES INSTITUTE ISI
213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CL IFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT This document is approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ISI/RR-89-199	5. MONITORING ORGANIZATION REPORT NUMBER(S) -------
6a. NAME OF PERFORMING ORGANIZATION USC/Information Sciences Institute | 6b. OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION -------
6c. ADDRESS *(City, State, and ZIP Code)* 4676 Admiralty Way Marina del Rey, CA 90292-6695	7b. ADDRESS *(City, State, and ZIP Code)* -------
8a. NAME OF FUNDING/SPONSORING ORGANIZATION RADC NSF | 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICAT ON NUMBER MCS-7918792 F30602-85-C-0221
8c. ADDRESS *(City, State, and ZIP Code)* --over--	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. ------- | PROJECT NO. ------- | TASK NO. ------- | WORK UNIT ACCESSION NO. -------

11. TITLE *(Include Security Classification)*
The Knowledge Representation Project(Unclassified)

12. PERSONAL AUTHOR(S) MacGregor, Robert and Yen, John.

13a. TYPE OF REPORT Research Report	13b. TIME COVERED FROM ______ TO ______	14. DATE OF REPORT *(Year, Month, Day)* 1989, July	15. PAGE COUNT 26

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
09	02	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
database management systems, knowledge representatio , termin logical reasoning classification (KR)

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

This report describes ISI's knowledge representation research, which can be divided into two phases. The initial phase produced the NIKL classifier, a system that provides a competent and efficient capability for inference applied to terminological knowledge. The second phase began with a study of how NIKL could be improved, and is now in the process of producing LOOM, a new knowledge representation system with significantly broader inference capabilities. This report describes these two current research efforts and discusses some future plans for ISI's research in knowledge representation. [illegible]

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Victor Brown Sheila Coyazo	22b. TELEPHONE *(Include Area Code)* 213/822-1511 | 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

(8c continued)

Air Force Systems Command,
Rome Air Development Center
Griffiss Air Force Base, NY 13441-5700

National Science Foundation
1800 G Street NW
Washington, DC 20550

ISI Research Report
ISI/RR-89-199
July 1989

University of Southern California

Robert MacGregor
John Yen

The Knowledge Representation Project

INFORMATION SCIENCES INSTITUTE
213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

This research is supported by the Air Force Systems Command, Rome Air Development Center under contract No.MCS-7918792, and by the National Sciene Foundation, under contract No.F30602-85-C-02201. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policy of ARDC, NSF, the U.S. Government, or any person or agency connected with them.

The Knowledge Representation Project at ISI

Robert Mac Gregor
John Yen

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

1. Introduction

ISI's knowledge representation research can be divided into two phases. The initial phase produced the NIKL classifier, which was the first system anywhere to provide a competent and efficient capability for inference applied to terminological knowledge. The second phase began with a study of how NIKL could be improved, and is now in the process of producing LOOM, a new knowledge representation system with significantly broader inference capabilities. This paper addresses these two efforts, beginning with a discussion of the need for adequate knowledge representation.

We begin by describing the features we require in a competent knowledge representation system. We argue that systems such as relational DBMSs or theorem provers do not meet these criteria, in-as-much as they provide only a part of the inferential capabilities needed by some of today's Artificial Intelligence (AI) applications. We then describe the architecture which has evolved in the family of knowledge representation systems which trace their ancestry back to the KL-ONE system, and introduce the notion of a *classifier*. Next, we will present more detail on the kind of technology that has grown up around classification-based knowledge representation systems. We explain what ISI's earlier contribution has been to this technology, and list some of the applications which use this technology. Finally, we describe the current research being carried out by ISI's knowledge representation project, and we describe some future plans for ISI's research in knowledge representation.

2. Competent Knowledge Representation Systems

The field of knowledge representation concerns itself with (1) providing a means for representing knowledge within a computer, and (2) providing mechanisms that can produce useful inferences based on the resulting knowledge structures.

The first examples of knowledge representation in AI were ad hoc systems constructed to meet the requirements of particular applications. In a typical system there were no well-defined semantics [Woods 75], and the knowledge was not *re-usable*, so that there was little that might be called a knowledge representation *technology*---knowledge representation consisted mainly in defining and manipulating data structures.

A principal argument found in the literature that has evolved since the advent of the KL-ONE knowledge representation language [Brachman and Schmolze 85] is that a *language* should be designed specifically for the purpose of representing knowledge. KL-ONE was designed to represent the kinds of knowledge constructs encountered by developers of natural language processing systems. The availability of a knowledge representation language allows its users the freedom to manipulate and question a knowledge base without having to be familiar with either the *data structures* used to represent the knowledge, or with the *inference mechanisms* that interpret requests from the user. An analogy can be drawn with the field of database management systems (DMBS). There, the advent of the relational model and relational query languages was heralded as a major improvement over the network models used in earlier database systems.

The advent of KL-ONE spawned a lot of concentrated activity in knowledge representation. Several high-level principles emerged, which strengthen our notion of the important components that constitute a competent knowledge representation system:

1. An expressive[1] and high-level language should exist for representing both terminological and assertional knowledge.[2]

2. The language should be accompanied by a rigorous semantic definition.

3. The system should provide competent inference mechanisms which respond to user requests about any knowledge entered into the system's knowledge base.

We will examine each of these criteria in turn as they apply to (1) a relational DBMS; (2) a logic-based system (e.g., Prolog or a generic theorem prover); and (3) one of the more robust descendants of KL-ONE.

---

[1] A highly expressive language places few restrictions on what kinds of knowledge can be represented, while a language with low expressivity allows only a few simple kinds of knowledge to be represented.

[2] *Terminological* knowledge consists of the definitions of the *terms* in some domain, e.g., the term **girl** might be defined as the conjunction of the terms **child** and **female**; *assertional* knowledge consists of statements about the way things are in the world, e.g., the statement **(girl Nancy)** asserts that the object denoted by the symbol **Nancy** satisfies the predicate **girl**. In a relational DBMS, the schema definitions represent terminological knowledge, while the tuples in the database represent assertional knowledge.

An Expressive High-Level Language

The degree of expressiveness provided by a relational DBMS is relatively low, and therefore such systems do not meet our first criterion. By contrast, a logic-based system using some variant of first-order logic (FOL) as its knowledge representation language does exhibit a high-degree of expressive power. Yet, [Brachman 82] and [Brachman, Fikes, and Levesque 83] argue that one can design languages that are more suitable than FOL for representing *terminological* knowledge. Languages such as KL-ONE, NIKL, and LOOM provide a rich syntax for defining terms. In each case, however, this syntax is not particularly appropriate for expressing assertional knowledge. Thus, the type of knowledge representation system that we advocate contains two sublanguages: a terminological language such as one of the languages just mentioned, and a separate assertional language assumed to be some subset of FOL.

A Rigorous Semantic Definition

The semantics of the terminological (schema) portion of a relational DBMS is almost non-existent, so it is rigorous by a vacuous argument. On the other hand, a great body of theoretical literature has discussed the semantics of the assertional component of a relational DBMS. Logic-based systems, of course, have an impeccably rigorous semantics for both the terminological and assertional components, up to the point where they introduce extra-logical operators (e.g., Prolog's "cut" operator), at which point they abandon their claim to rigor.

Several rigorous semantic definitions have been worked out for KL-ONE-like languages. [Schmolze 85] describes a set-theoretic semantics for NIKL; [Brachman, Fikes, and Levesque 83] describes a semantics for KRYPTON based on logical entailment. Appendix A illustrates the semantics for the terminological component of LOOM.

Competent Inference Mechanisms

A relational DBMS possesses a query facility that is able to answer queries about its assertional knowledge. However, it has no capabilities whatsoever for reasoning with terminological knowledge. A logic-based system's ability to reason with assertional knowledge is in general very good (ignoring possible questions of efficiency). However, they both have trouble reasoning with terminological knowledge. There are two sources of difficulty. First, a terminological definition is equivalent (from a reasoning standpoint) to a universally-quantified bi-conditional proposition. Logic systems traditionally have difficulties in dealing with bi-conditionals (e.g., they cannot even be expressed in Prolog).[3] Second, some of the questions we would like to have answers to (e.g., what other terms are implied by the term "battle-situation") are second-order

[3]From a formal standpoint, the Prolog language does not possess a capability for defining terms.

questions that are generally beyond the scope of today's theorem provers.

The *classifier* is an inference mechanism developed specifically for answering questions about analytic relationships between terms. Its principal function is to compute *subsumption* relationships between terms.[4] In addition to providing question-answering capability, a classifier is typically used to organize all of the terms defined for a knowledge base into a taxonomy in which more general terms are placed above more specific ones. This capability for self-organization is one of the most striking features of this type of system. Section 6 lists a number of applications where the inferential capabilities of a classifier have proven to be useful.

Hybrid systems, which utilize a classifier to reason with terminological knowledge, and which possess a separate (more traditional) inference mechanism for reasoning with assertional knowledge, are the only systems in use today that demonstrate inferential competence for *both* terminological and assertional knowledge. We expect that as the ability to reason with "meta-level" (terminological) knowledge becomes less of a dream and more of a reality in AI systems, the demand will increase significantly for knowledge representation systems that display inferential competence at the terminological level. Appendix B is a matrix containing a feature analysis that summarizes our evaluation of four classes of knowledge representation systems.

3. Classification-based Technology

This section describes more of the technology now associated with those knowledge representation systems that center their inference capabilities around a classifier. We review some of the theoretical results derived during the evolution of this technology, and describe the impact that these results have had on the design of practical systems. This section concludes with an examination of the criteria used to determine where the line that divides the TBox and ABox knowledge spaces should be drawn.

The component of a system that deals with terminological knowledge is commonly called a "TBox," while the component that manages assertional knowledge is called an "ABox." Clearly, it is necessary that some facility be provided which "bridges the gap" between the TBox and the ABox. Classification-based hybrids incorporate a component called a *recognizer* (also called a "realizer"), which serves to relate ABox knowledge to TBox knowledge (see [Vilain 85]). A recognizer is the dual of a retrieval mechanism: Given a (TBox) concept, a retrieval mechanism is able to find all (ABox) instances belonging to the extension of that concept; given an ABox instance, a recognizer is able to produce all TBox concepts that include that instance in their extensions (i.e., all

[4] A term A *subsumes* another term B if instances (members of the extension) of B must also be instances of A.

concepts that *describe* that instance).[5] More prosaically, a recognizer can answer questions such as "Tell me everything you (the system) know about John Q. Public," or "Tell me what you know about the current status of this steam turbine."

Two distinct reasoning components, a *classifier* and a *recognizer*, form the basis of the architecture of our hybrid systems. Typically, the recognizer utilizes the classifier to perform the most difficult (from an engineering standpoint) of its inferences, so the scope and power of the over all system depends most heavily upon the abilities of its classifier. Originally, designers of classifiers intended that the mechanism for testing subsumption relationships between TBox concepts should be *sound*, *complete*, and should run in polynomial time, i.e., it should be *tractable*. Unfortunately, theoretical analyses of various candidate TBox languages [Brachman and Levesque 84], [Patel-Schneider 87] has revealed that tractability can only be achieved for *very* restricted ("impoverished") TBox languages. These theoretic results guarantee (unless P=NP) that the desirable goals of *soundness*, *completeness*, *tractability*, and a reasonably high degree of *expressivity* (in the TBox) cannot be simultaneously achieved.

Systems that are both principled and reasonably efficient require soundness and tractability to be retained, and thus the design choice to be made is a trade-off between completeness and expressivity. Most of today's classifiers have opted to sacrifice completeness in favor of expressivity because of the real demands placed on knowledge representation by today's intelligent systems. It is generally more useful to have a system capable of providing *some* information in a timely manner than to have a guarantee that you will get a complete answer *eventually.*

With this brief background, we can now address the question "How do we decide what knowledge goes in the TBox, and what goes in the ABox, i.e., where should we draw the line?" The answer has an empirical basis, rather than a theoretical or philosphical one: We desire a fairly expressive TBox language. On the other hand, we desire that users of a classification system can "trust" it to find what they regard as all "reasonable" inferences (while realizing that our TBox classifier will miss some inferences (classifications) due to the fact that it is necessarily incomplete). Put another way, within the limits established by the degree of expressiveness of the TBox language, the classifier should be "as smart as its users."

There has been a good deal of experimentation to find a satisfactory set of primitives (together with a set of algorithms for reasoning with those primitives). Within the community, a consensus has been reached on the choice of the more basic operators. However, these experiments have tended to stay within the bounds of expressivity first established by KL-ONE. One of LOOM's major contributions is to demonstrate the successful incorporation of several new primitives which significantly

[5]This statement is not quite accurate---if the classifier is not *complete*, then some inferences may be missed, i.e., some, but not **all**, concepts will be found.

extend the language's expressivity while maintaining overall inferential competence and uniformity.

4. An Example

To make this discussion clearer, we present an example that illustrates one possible application of a classifier. We first sketch out the problem, which is to construct a model of a nuclear reactor, and invent a partial methodology for how the model is to be constructed. Next, we run through a scenario containing a few fragments of such a model. Finally, we discuss the broader implications of this approach to model-building, and provide some commentary on the type of inferences exhibited by the classifier.

Our model of a reactor has been constructed by first building a reactor model consisting of abstract descriptions of its various components, and then refining or instantiating the components. In our scenario, one or a team of more experienced designers have initially created a knowledge base containing (1) fully-abstract concepts such as "reactor part" and "pressurized fluid"; (2) "guidance rules," which help to guide partially defined concepts into their proper place in the model; and (3) "safety rules," which can be invoked to verify that newly defined components obey specific safety criteria. We imagine that the initial model is turned over to a team of (possibly less-experienced) modellers who will develop an increasingly accurate design of an actual reactor. The rules defined up-front help to insure the consistency and correctness of the system built during this second phase.

Below is a fragment of a model, expressed in the LOOM language.[6] Below that is an English explanation of each of the LOOM constructs.

```
;; Model of Reactor:
[1] (defconcept Reactor-Part)
[2] (defconcept Pressurized-Part
        :is (:and Reactor-Part (:exactly 1 psi))
        :constraints (:exactly 1 max-rated-psi))
[3] (defconcept Pressurized-Fluid)
[4] (defconcept Reactor-Coolant :is (:and Pressurized-Fluid :primitive))

;; Guidance Rules:
[5] (implies
        (:and Reactor-Part (:the type-of-fluid Pressurized-Fluid))
        Pressurized-Part)

;;Safety Rules:
[6] (defconcept :is Reactor-Part
        :disjoint-covering (Safe-Part Unsafe-Part))
[7] (implies
        (:and Pressurized-Part Safe-Part)
```

[6]Note: Appendix A summarizes the LOOM constructs used to synthesize concept definitions.

```
       (<= psi max-rated-psi))
[8] (:implies (:and Safe-Part
                    (:the type-of-fluid Reactor-Coolant))
       (>= psi 10))

;; Description of Specific Part:
[9] (defconcept Reactor-Outlet-Piping :is (:and Reactor-Part :primitive)
       :constraints ((= max-rated-psi 5)
                     (:the type-of-fluid Reactor-Coolant)))
```

English explanation of this model:

[1] There is a concept called a "reactor part";
[2] X is a "pressurized part" IF-AND-ONLY-IF X is a reactor part and X has the attribute "psi";
IF X is a pressurized part THEN X has an attribute "max-rated-psi";
[3] There is a concept called a "pressurized fluid";
[4] A "reactor coolant" is a particular kind of pressurized fluid;

[5] IF X is a reactor part and the value of "type-of-fluid" for X is "pressurized fluid" THEN X is a pressurized part;
[6] A reactor part is either a safe part or an unsafe part, but not both;
[7] IF X is both a reactor part and a safe part THEN
the value of X for "psi" is not greater than its value for "max-rated-psi";
[8] IF the value of "type-of-fluid" for X is "reactor coolant" and X is a safe part THEN the value of "psi" for X is at least 10;

[9] A "reactor-outlet-piping" is a particular kind of reactor part;
IF X is a reactor-outlet-piping THEN the value of "max-rated-psi" for X equals 5, and the value of "type-of-fluid" for X is "reactor coolant";

4.1. Defining the Concept "Reactor-Outlet-Piping"

Suppose we are in a state where concepts [1]-[8] are in place, and our job is to define a concept representing a piece of piping that will carry reactor coolant away from the reactor. The final description will have a large number of attributes describing its size, where it gets attached, etc., but initially we just wish to sketch in a few features. So, we define the concept [9] above, naming it "Reactor-Outlet-Piping". Now the classifier goes to work:

Because Reactor-Outlet-Piping has "type-of-fluid" set to Reactor-Coolant, which is a type of Pressurized-Fluid, it gets classified below the (unnamed) concept which is the first argument of the implication [5]. Applying the implication in [5] to Reactor-Outlet-Piping reveals that instances of Reactor-Outlet-Piping are necessarily instances of Pressurized-Part. At this point, we may or may not elect to redefine Reactor-Outlet-Piping to specialize Pressurized-Part explicitly.

The model builder's manual indicates that we should periodically check the safety of our designs. We do this for our concept Reactor-Outlet-Piping by determining whether or not an instance of Reactor-Outlet-Piping can also be an instance of Safe-Part, i.e., we construct and classify the following definition, and then test to see if it has a non-null extension.

```
[10]  (:and Reactor-Outlet-Piping Safe-Part)
```

Concept [10] will classify the concepts [7] and [8], and hence will inherit both of the constraints contained in those concepts. Combining those constraints with the originally defined restriction on the attribute "max-rated-psi" yields a contradiction. The classifier therefore marks the resulting concept as "incoherent." The result after classification is the concept:

```
[11] (defconcept
        :is (:and Reactor-Outlet-Piping Pressurized-Part Safe-Part)
        :constraints ((>= psi 10)
                      (= max-rated-psi 5) (<= psi max-rated-psi)
                      (:exactly 1 psi) (:exactly 1 max-rated-psi)
                      (:the type-of-fluid Reactor-Coolant)
                      :is-incoherent))
```

Based on the constraints implied by this concept, the classifier will have calculated that "psi" is both greater than 10 and less than 5, which is impossible. We have therefore determined that our description of Reactor-Piping-Outlet is incompatible with Safe-Part.

4.2. Discussion of the Example

Several points are note-worthy:

- A striking feature of this example is that all of the inferences performed by the classifier were applied to *descriptions*, i.e., all reasoning took place at the "meta-level". As far as we know, all of the tools available today that might be applied to such a problem (e.g., production systems, frame-based-systems, relational-dbms-based systems, Prolog) are capable of reasoning *only with instantiated (ground-level) data.*

- The knowledge base used in this example is *fully-declarative*. Essentially, it just represents an encoded form of a collection of axioms in the predicate calculus. The ability of the system to perform analysis at the meta-level depends on the fact that all of the knowledge is represented explicitly.

- The example illustrates a classifier operating by embedding new concepts into an existing taxonomy. This exemplifies its capability for *self-organization* of knowledge.

- The reasoning described illustrates *synthetic*, as opposed to *analytic* reasoning. During our classifications, additional information about Reactor-Outlet-Piping was acquired from the rules [5], [7], and [8], whereas purely analytic reasoning, by definition yields no new information about a concept. The LOOM classifier, when completed, will be the first classifier capable of synthetic reasoning.

- Finally, our example nicely illustrates how statically defined constraints can be used to guide the model-building process.

5. The NIKL System

The original KL-ONE system successfully demonstrated "proof of concept," but it ran much too slowly to be used in applications. A joint effort by researchers from ISI and BBN produced a design for a new terminological language called NIKL. ISI then produced a classifier for the NIKL language that was very much faster than the KL-ONE classifier -- fast enough to be a valuable research tool. The benefits were immediately realized as KL-ONE-based applications at ISI and BBN were converted to use NIKL.

Subsequently, ISI made few modifications to the NIKL system until DARPA funded an new ISI project called Empirically Valid Knowledge Representation in 1986. One of the first tasks of the new project was to translate NIKL into Common LISP -- a move that significantly increased its accessibility to the research community. Accordingly, an increasing volume of requests for NIKL have come from around the world. Appendix C lists 17 institutions that have solicited and received a copy of NIKL.

Another task of the new project produced a graphic display capability for viewing the concept networks produced by the NIKL classifier. The "ISI Grapher," which was designed as a general-purpose tool, has had a fairly spectacular reception outside of ISI. A paper on the Grapher [Robins 87] has appeared as an invited talk in Symboliikka '87, a Finnish conference on user-interfaces and graphics, while 178 sites have expressed interest in obtaining the ISI Grapher. The Grapher has been shipped to 50 of these sites. This work does not represent deep research, but it does represent work that has practically realized research ideas. Furthermore, the results are of direct benefit to the research community, and have been applied in practical, technological implementations.

6. Examples of the Use and Influence of NIKL

The development of NIKL led to a rapid increase in the number of research efforts using KL-ONE-like knowledge representation languages. The Consul user interface management effort was the first to employ the language. The basic user interface used the models to structure displays [Mark 81], and to supply semantics to terms used in a forward chaining inference system. Consul also used NIKL in a natural language case frame parser where the model represented some of the lexical semantics, as well as the case frames [Sondheimer 84]. User interface research has continued with the use of

NIKL in a multi-modal user interface system, II [Arens 87]. Natural language research has seen NIKL applied in a text generation system [Sondheimer 86] and another natural language understanding system [Weischedel 87]. In the two natural language efforts, the systems were able to share a common world model through a NIKL knowledge base.

Parallel applications have occurred in the area of expert systems. The Explainable Expert System effort used NIKL to describe the problem domain in an effort that produced a expert system creation system. [Neches 85] Another effort, BACKBORD, is producing a browsing interface for databases or knowledge bases, which utilizes NIKL knowledge bases to help users who are searching for information but need assistance in formulating a request that will retrieve exactly what they are looking for. [Yen 87] A related effort, TINT, adds a notecard facility to NIKL models as a mechanism for reducing the "brittleness" of expert systems when approaching the boundaries of their knowledge. [Harp 87]

All these applications have occurred at ISI and BBN. ISI has also interacted with graduate students and professors who are using NIKL in their research at institutions such as Massachusetts Institute of Technology, Carnegie Mellon University and the University of Pennsylvania.

One of the healthiest signs of NIKL's success is the number of researchers who have used NIKL as a starting point for their work. KL-TWO uses NIKL as a T-Box to which it added an A-Box [Vilain 85]. Brachman has related that NIKL is the standard by which he has measured the Krypton T-Box [Brachman 85]. In part because it was impossible to send NIKL outside the United States for many years, there have been many European imitators. Most notable are the BACK system, which shows a well-matched T-Box and A-Box [von Luck 87] and SB-ONE, which has benefited from extensive development. [Xtra 87].

7. The LOOM System

The Empirically Valid Knowledge Representation project (introduced in Section 5) came into existence because users of NIKL had generated a substantial list of requests for fundamental improvements and extensions to NIKL (see [Kaczmarek 86]). The requests were for:

- an Incremental Classifier -- users wanted to be able to modify the definitions of already-classified concepts, and wanted the classifier to reclassify all concepts impacted by each modification.

- an ABox and Recognizer -- while NIKL provides a competent and efficient TBox component, there was no ABox or recognizer available with comparable capabilities.

- major Extensions to the Terminological Language:

- a richer vocabulary for defining binary relations, e.g., users wanted to be able to define inverse, transitive-closure, and composed relations;
- specialized representations for Sets, Intervals, and Sequences;
- a capability for representing necessary conditions and sufficient conditions (*constraints*).

Considerable research went into the problem of introducing *constraint* knowledge into the classification paradigm. The solution required that the semantic basis of NIKL be revised in certain places. To indicate that significant changes were made, the new system was given a new name -- LOOM.

In section 7.1, we give an account of the inference mechanism developed to reason about constraints in LOOM. This illustrates some of the research that has gone into the design of the LOOM system.

7.1. The CBox

As already mentioned, KL-ONE-based systems have an established tradition of making a strong distinction between *terminological* (TBox) knowledge and *assertional* (ABox) knowledge. In order for the classifier to reason with constraints, we found it necessary to further partition the knowledge in the ABox. Assertions about *classes* of individuals will be labelled *constraint* knowledge and placed in a *CBox,* while assertions about *single* individuals will remain in the ABox.

The most useful type of constraint takes the form of an *implication*, i.e., it has the form "IF X is an instance of the class P THEN X is also an instance of the class Q". In our previous examples, all uses of the **implies** operator or the **:constraints** keyword represented specifications of implication relationships, e.g. "IF X is a pressurized part THEN X has an attribute 'max-rated-psi'".[7] We have developed a new type of classifier, called a *CBox classifier,* which is able to compute implication relationships between concepts based on both definitional and constraint knowledge. (The inferences illustrated in the Reactor Part example were mostly the result of CBox classifications, rather than TBox classifications). Basic to a traditional classifier's operation is that it computes the *subsumption* relationships between all pairs of concepts in a network. The CBox classifier expands this paradigm by computing *implication* relationships between all pairs of concepts in a network. Thus, for example, it is easy for it to answer the "second-order" question "What terms are implied by the term 'battle-situation'?"

The language used to express TBox knowledge is deliberately restricted so as to

[7]The CBox contains other knowledge besides implications (e.g., assertions of disjointness). This falls outside of the scope of our present discussion.

exclude certain constructs (e.g., recursive definitions) that are not amenable to classification. We repeat this technique for constraints -- the syntactic structures that appear in LOOM **:constraints** or **implies** clauses translate into knowledge structures for which we have developed competent classification algorithms. As we will illustrate below, the CBox classifier utilizes the TBox classifier to perform most of its inferences. Thus, not only is the syntax for implications similar to that for TBox knowledge (as illustrated in the previous section), but the same *data structures* can be used to represent both types of knowledge. This has the practical benefit that each new inference capability added to the TBox classifier automatically extends to the CBox classifier as well.

Here, we sketch the algorithm for CBox classification:[8] the computation of each implication relationship is represented internally by an "implies" link that links a concept C to a concept CI which represents the conjunction of all concepts implied by C. (Before C is classified, C's definition may indicate a *set* of concepts that are implied by C).

CBox Classification Algorithm:

First, tbox-classify C. Define C1 to be the conjunction of all concepts reachable from C by following implies links (i.e., compute a transitive closure over the implies links). Tbox-classify C1. Define C2 to be the conjunction of all concepts implied by C1 and then tbox-classify C2. Repeat until C(k) = C(k+1). Set CI = C(k).

A key element in the efficiency of this algorithm is that deductions made during previous cbox-classifications are completely characterized by the "implies" links placed in the network, with the result that no time is wasted recomputing results which were deduced previously. Because the implies relation is reflexive, it is necessarily the case the C(i) subsumes C(i+1) for all i's. Hence, termination of the algorithm is guaranteed.

It may not be obvious that the classification cycle needs to be repeated, i.e., that k > 1. However, a trace of the algorithm applied to our Reactor-Part example reveals that (depending on the order in which the network is traversed) two complete classifications may be necessary to determine that the concept [10] implies the concept [7]. We have produced artificial examples that show that k can be arbitrarily large; however, we expect that, for real applications, it will usually turn out that the second classification will yield no new information. Hence, we will be looking for heuristics that recognize situations for which a single classification suffices.

[8]The CBox classification algorithm implemented in LOOM is more efficient (and more complex) than the one sketched here, but the overall inference *strategy* is the same.

8. Future Plans for Knowledge Representation Research

We describe two different research directions that are planned by ISI's Knowledge Representation Project. First, we describe plans for future extensions to LOOM itself. Second, we describe work on a new style of programming that uses a classifier as its basic control mechanism. This effort is a part of the Shared Knowledge Representation project, which is a follow-on to the Empirically-Valid Knowledge Representation project.

8.1. Future Work on LOOM

We envision a shift in the character of future extensions to LOOM, away from the present emphasis on abstract, mathematical knowledge structures, and towards modelling and reasoning about physical-world knowledge. Some immediate candidates for modelling include Time, Location, Events, Actions, Change of State, and Collectives. The starting point for some of these will be models already developed by other projects within ISI.

There are two major guidelines that will constrain our modelling efforts. The first is that whatever we produce must be, to the greatest practical extent, application-independent. For example, we would expect that each of the other projects will have its own notion on the best way to model Time. We will have to create a model that is simultaneously acceptable to all of the different internal projects. The fact that at least six other projects internal to ISI will be using LOOM in the near future offers a unusual opportunity: a consensus among these projects relative to some construct should stand a reasonable chance of carrying over to applications outside of ISI. The payoff, of course, is that as these projects increase their use of commonly developed knowledge structures, the possibilities increase for inter-communication and sharing of knowledge across applications.

The second major modelling challenge is that each new modelling construct must be made to fit into the classification paradigm. For example, here is how we have already chosen to handle numeric comparisons:

> Suppose we introduce two new concepts: "P50" is the set of Persons of age at least 50, and "P60" is the set of Persons of age at least 60. LOOM will create two "intervals" [50..Infinity) and [60..Infinity), classify the intervals, discover that [50..Infinity) subsumes [60..Infinity), and then conclude that P50 subsumes P60.

In the above example, LOOM handled numeric comparisons by converting them to numeric intervals, and then handing the problem over to a reasoner that understands intervals. The point here is that adding a new capability to LOOM involves more than just agreeing on a few data structures. A precise semantics must be formulated, usually, some special-purpose algorithms must be developed; and care must be taken that new primitives are orthogonal to the existing ones. (Internally, the classifier relies on achieving a canonical representation for its knowledge structures -- this requirement is defeated if redundancy or overlap creeps into its set of primitive constructs.)

The resulting payoff is that a careful implementation of a knowledge structure will allow LOOM to reason competently and efficiently in that new domain, and it will allow that new reasoning capability to blend-in smoothly with other specialized reasoners. The classification paradigm appears to provide a very good medium for installing and integrating multiple domain-specific reasoners.

8.2. Classification-based Programming

There have already been systems written (within the CONSUL [Mark 81] and EES [Swartout and Neches 86] projects) that demonstrate how the control portion of a program can be implemented with a classifier.[9] Such programs are "classification-based." The increased expressive power of LOOM allows us to encode a greater percentage of our program in terms of LOOM constructs -- the goal is that the "core" of an application program can be coded entirely within a LOOM-like language. The long-range goal of this research is to produce a new programming technology superior to the rule-based technology in use today.

In this section, we briefly characterize the kinds of inferences made by the LOOM classifier, and then discuss how we are planning to employ this reasoning capability within the context of a general programming environment.

Cast into a logic framework, the definitions found in a TBox map into bi-conditional propositions. Thus, a classifier's forte is reasoning with bi-conditionals -- something that many logic-based systems (e.g., Prolog) find difficult or impossible. A major achievement of the LOOM system is that it incorporates ordinary conditionals into the classification framework (where they are referred to as "implications"). The inferences drawn by the LOOM classifier are all "forward" inferences -- whenever new knowledge is introduced into the system, LOOM's response is to immediately compute all other new propositions implied by that new bit of knowledge. The results of these computations are most often preserved as semantic links placed between nodes in the network, where each link represents an instance of a deduced relationship. (LOOM's repertoire currently includes subsumption, implication, inverse, and transitive-closure links).

We expect that this characterization of LOOM's reasoning capabilities will remain invariant as LOOM evolves to incorporate additional modes of special-purpose reasoning. In particular, while the reasoning power of LOOM will continue to increase, the system will never on its own achieve Turing-completeness. Therefore, to *program* with LOOM, it is necessary that a Turing-complete language (e.g., LISP) be interfaced with LOOM, resulting in a hybrid programming language.

[9]The NIKL classifier was used in both of these projects.

Conceptually, the LOOM recognizer is designed to react immediately to each change in its knowledge base, and it is able to determine (modulo incompleteness) all consequences of each change. We expect to find this behavior valuable for such applications as discrete simulation and process control. Thus, while current applications of *classification technology* all lie within the domain of AI, we expect other non-AI applications will benefit as well once the technology has reached a sufficent state of maturity.

The LOOM recognizer can be viewed as a powerful and efficient pattern matcher that matches a new instance (datum) with all concepts (patterns) in T-box. It is powerful because the matching process is based on the semantics, rather than the syntax, of the data and the patterns. The recognizer is efficient because (1) the patterns are organized into a taxonomy (analogous to the RETE pattern nets existing production systems), and (2) the recognizer only needs to consider those patterns that are semantically relevant to the datum. Recently, we have produced (in conjunction with the ISI's FAST and EASES projects) a preliminary design for a classification-driven production system that uses the LOOM recognizer as its pattern matcher.

The architecture of a *classification-triggered* production system differs from conventional one in that the system is triggered by new classifications rather than by the recognize-select-execute cycles. The condition of each production is represented as a class. Thus, when an instance gets classified under a class, all productions whose conditions are the class get instantiated and passed to an external *production manager* that selects and executes the instantiated productions.

The future work of this research includes (1) defining a language to express production rules, (2) specifying the interface between LOOM and the production manager (which will be designed and implemented under the FAST and EASES projects), and (3) implementing the productions in LOOM. As a test bed, the FAST project will apply the production system to building an expert system for recognizing bad part numbers. Eventually, integrating the reasoning capabilities of LOOM into the production system architecture will generate a new programing environment that facilitates representation, reasoning, and acting on various kinds of knowledge in AI systems.

9. Summary

We have presented an informal standard for evaluating the competence of a knowledge representation system. Our most stringent criterion is that a knowledge representation system *should demonstrate inferential competence* for both terminological and assertional knowledge.

The classification-based knowledge representation systems that trace their ancestry back to KL-ONE are developing a promising technology to yield systems that exhibit inferential competance and are efficient enough to be used in practical applications. The classification paradigm is proving to be a good medium for embedding multiple efficient, domain-specific reasoners.

ISI is playing a major role in advancing the state of the art in classification-based systems, and in delivering practical implementations of this technology. The NIKL classifier provided the first example of a practical terminological reasoner. Research that has gone into LOOM, the successor to NIKL, has considerably widened the scope of inferences that can be captured within the classification paradigm. Among LOOM's innovations will be the ability to compute all implication relationships between terms in a taxonomic network. LOOM will exhibit significantly more comprehensive inferential capabilities than its predecessor, while retaining the efficient classification algorithms that went into the NIKL system. LOOM will find immediate application in a number of on-going research projects at ISI, and has sparked the interest of a number of AI research sites outside of ISI.

A. LOOM Semantics

Summary of LOOM Expression Operators

Connectives

`(:and` C_1 ... C_j`)`	the conjunction of concepts/relations C_1 ... C_j
`(:or` C_1 ... C_j`)`	the union of concepts/relations C_1 ... C_j
`(:not` C`)`	the complement of the concept/relation C

Concept Expressions

`:primitive`	a unique primitive concept
`(:at-least` k R`)`	the role R has at least k values
`(:at-most` k R`)`	the role R has at most k values
`(:exactly` k R`)`	the role R has exactly k values
`(:all` R C`)`	all values of the role R have type C
`(:some` R C`)`	at least one value of the role R has type C
`(:the` R C`)`	role R has exactly 1 value, and it has type C
`(:same-as` R_1 ... R_j`)`	roles R_1 ... R_j have identical values
$(REL\ R_1\ ...\ R_j)$	the relation/operator REL is satisfied by the values of the roles R_1 ... R_j

Relation Expressions

:primitive	a unique primitive relation
`(:domain` C`)`	the domain fillers have type C
`(:range` C`)`	the range fillers have type C
`(:inverse` R`)`	the inverse of the relation R
`(:lambda` (*args*) . *body*`)`	a (primitive) operator relation

Relation Attributes (for a relation R)

`:single-valued`	$(R(x,y) \wedge R(x,z))$ implies $y = z$
`:closed-world`	closed-world semantics apply to R's role fillers
`:symmetric`	$R(x,y) = R(y,x)$
`:sequence`	R's role fillers form a sequence

Set Expressions

`(:symbols` S_1 ... S_j`)`	the set of symbolic literals $\{S_1\ ...\ S_j\}$
`(:instances` I_1 ... I_j`)`	the set of database instances $\{I_1\ ...\ I_j\}$

Interval Expressions

`(:through` S_1 ... S_j`)`	the set of scalars between S_1 and S_j, inclusive

B. Feature Analysis

FEATURE ANALYSIS OF CANDIDATE KNOWLEDGE REPRESENTATION SYSTEMS

	Feature \ System	Relational DBMS	Expert System Shell	Hybrid Classification-Based System	FOL Theorem Prover
ASSERTIONAL COMPONENT	Language with well-defined semantics	Yes	Yes	Yes	Yes
	Expressiveness of Language	Low	High	Medium-High	High
	Inference Capabilities	Medium	High	High	Very High
	Efficiency	Very High	Medium	Medium	Low
TERMINOLOGICAL COMPONENT	High-Level Language	No	No	Yes	No
	Well Defined Semantics	Yes	No	Yes	Yes
	Expressiveness of Language	Very Low	Low	High	High
	Inference Capabilities	None	None	High	High
	Efficiency	N/A	N/A	Medium	Very Low
	Second-Order Reasoning	No	No	Yes	No

C. NIKL Users

NIKL has been installed at the organizations listed below:

University of Southern California, ISI
Bolt, Beranek, and Newman, Boston, MA
Massachusetts Institute of Technology
Carnegie-Mellon University
Lockheed AI Center Menlo Park, CA
Courant Institute of Math. Sci., New York University, New York
MCC, Austin Texas
The MITRE Corporation, Bedford, MA
San Francisco State University
University of Florida, Gainesville, FL
University of Pennsylvania, Philadelphia, Penn.
St. Patrick's College, Dublin, Republic of Ireland
Technische Universitat Berlin, Federal Republic of Germany
UNISYS, Paoli, Penn.
University of Leeds, England, United Kingdom
University of Saarbruecken, Federal Republic of Germany
Whitney/Demos Productions, Culver City, CA

References

[Arens 87] Yigal Arens, Lawrence Miller, Norman Sondheimer, Presentation Planning Using an Integrated Knowledge Base, 1987. Submitted for publication.

[Brachman 82] R.J. Brachman and H.J. Levesque, "Competence in Knowledge Representation," in *Proceedings of AAAI-82, The National Conference on Artificial Intelligence*, AAAI, Pittsburgh, PA, August 1982.

[Brachman 85] R. J. Brachman, V. P. Gilbert, H. J. Levesque, "An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 532-539, Los Angeles, CA, August 1985.

[Brachman and Levesque 84] Ronald J. Brachman and Hector J. Levesque, *The Tractability of Subsumption in Frame-Based Description Languages*, Fairchild Research Laboratories, Technical Report, 1984.

[Brachman and Schmolze 85] Brachman, R.J., and Schmolze, J.G., "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science*, August 1985, 171-216.

[Brachman, Fikes, and Levesque 83] Ronald Brachman, Richard Fikes, and Hector Levesque, "KRYPTON: A Functional Approach to Knowledge Representation," *IEEE Computer*, September 1983.

[Harp 87] Harp, B. & Neches, R., A Knowledge-based Notecard Environment, 1987. Paper submitted to CHI'88.

[Kaczmarek 86] T. Kaczmarek, R. Bates, G. Robins, "Recent Developments in NIKL," in *AAAI-86, Proceedings of the National Conference on Artificial Intelligence*, AAAI, Philadelphia, PA, August 1986.

[Mark 81] William Mark, "Representation and Inference in the Consul System," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, IJCAI, 1981.

[Neches 85] Robert Neches, William R. Swartout, and Johanna Moore, "Explainable (and Maintainable) Expert Systems," *IEEE Transactions on Software Engineering* SE-11, (11), 1985, 1337-1351.

[Patel-Schneider 87] Peter F. Patel-Schneider, *Decidable, Logic-Based Knowledge Representation*, Schlumberger, Technical Report 56, May 1987.

[Robins 87] Gabriel Robins, "The ISI Grapher: a Protable Tool for Displaying Graphs Pictorially," in *Symboliikka '87*, Helsinki, Finland, August 1987.

[Schmolze 85] James G. Schmolze, The Language and Semantics of NIKL, 1985.

[Sondheimer 84] Norman K. Sondheimer, Ralph M. Weischedel, and Robert J. Bobrow, "Semantic Interpretation Using KL-ONE," in *Proceedings of Coling84*, pp. 101-107, Association for Computational Linguistics, July 1984.

[Sondheimer 86] Norman Sondheimer, Bernhard Nebel, "A Logical-Form and Knowledge-Base Design for Natural Language Generation," in *AAAI-86, Proceedings of the National Conference on Artificial Intelligence*, AAAI, Philadelphia, PA, August 1986.

[Swartout and Neches 86] William Swartout and Robert Neches, "The Shifting Terminological Space: An Impediment to Evolvability," in *AAAI-86, Proceedings of the National Conference on Artificial Intelligence*, AAAI, Philadelphia, PA, 1986 1986.

[Vilain 85] M. Vilain, "The Restricted Language Architecture of a Hybrid Representation System," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 547-551, Los Angeles, CA, August 1985.

[von Luck 87] K. von Luck, B. Nebel, C. Peltason, A. Schmiedel, *The Anatomy of the BACK System*, Technische Universitat Berlin, Technical Report KIT Report 41, January 1987.

[Weischedel 87] R. Weischedel, D.Ayuso, A. Haas, E. Hinrichs, R. Scha, V. Shaked, and D. Stallard, *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program*, BBN Laboratories Incorporated, Cambridge, MA, Technical Report Report No. 6522, June 1987.

[Woods 75] William A. Woods, *Whats's in a Link: Foundations for Semantic Networks*, Academic Press, 1975.

[Xtra 87] XTRA Group, *XTRA Progress Report 1985-87*, University of Saarbruecken, Saarbruecken, West Germany, Technical Report SFB 314, 1987. In German

[Yen 87] Yen, J. & Neches, R., Retrieval by Reformulation in a Multi-Purpose Browsing Interface, 1987. Paper submitted to CHI'88.